Spark Project

Name: Mohamed Ahmed Fathy

Business concept behind the project :

ITI want to develop the job fair process and want to target international companies for student to work abroad to increase the number of people working abroad to solve the dollar issue, they want a fact table to be made over twitter to know all the tweets that are recruiting data engineers all over the world for the next 9 months, the script need to run every 10 minute and get real time data of the last 10 minute tweets talking about data engineer .

Plan in the project:

ITI isn't interested on all tweets that has word data engineer, so we need to filter on the tweets that included Job_Alert or Hiring word beside data engineer, first made several scripts to be run in 1 script , and that script with cron job will run every 10 minute and should retrieve the tweets for last 10 minutes.

List of scripts : [twitter_pipline.py , twitter_listener.py , create_dimension.hql , twitter_stream.py , load_tweet_data.hql , spark_job.py]

Note: this code runs every 10 minutes, streaming not always opened, and in spark I make 2 tables separate, 1 in HDFS with all data, and in hive landing table which has only the new ones that we insert in dimension, there will be another solution in the folder with opened streaming and hive table on top of hive

Let's take a look on each script :

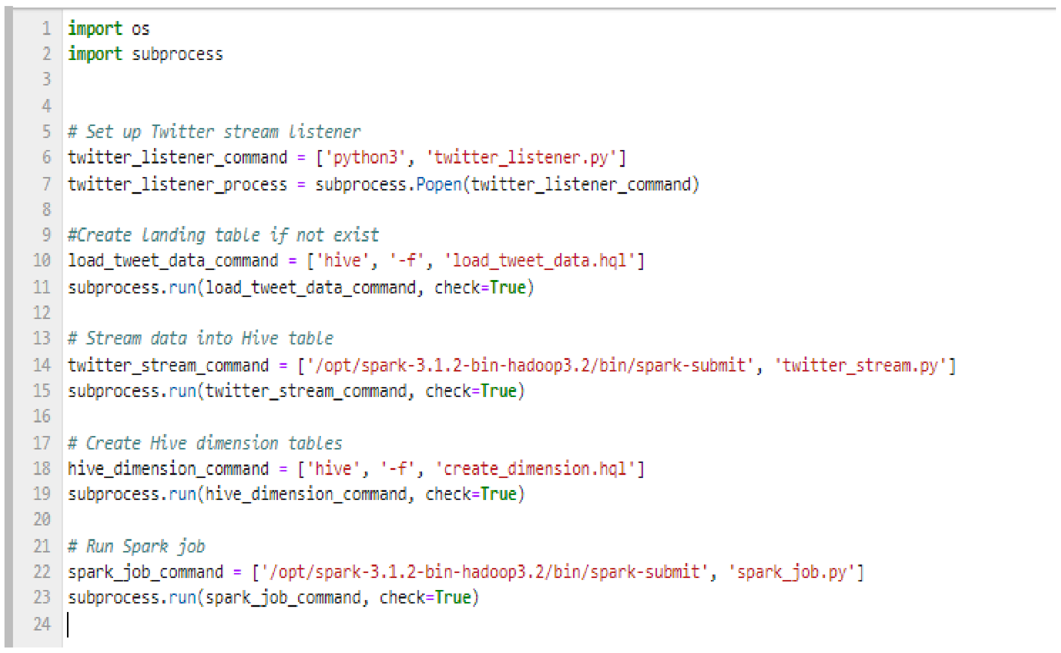
1- **twitter_pipline.py**

first to run it : python3 twitter_pipline.py

that is just a script that lists all other scripts inside it to run it automatically when we put it in cron job or airflow

note: twitter_listener must run in another terminal because its blocking script that's why we use .Popen

screen of it :



```python
import os
import subprocess


# Set up Twitter stream listener
twitter_listener_command = ['python3', 'twitter_listener.py']
twitter_listener_process = subprocess.Popen(twitter_listener_command)

#Create landing table if not exist
load_tweet_data_command = ['hive', '-f', 'load_tweet_data.hql']
subprocess.run(load_tweet_data_command, check=True)

# Stream data into Hive table
twitter_stream_command = ['/opt/spark-3.1.2-bin-hadoop3.2/bin/spark-submit', 'twitter_stream.py']
subprocess.run(twitter_stream_command, check=True)

# Create Hive dimension tables
hive_dimension_command = ['hive', '-f', 'create_dimension.hql']
subprocess.run(hive_dimension_command, check=True)

# Run Spark job
spark_job_command = ['/opt/spark-3.1.2-bin-hadoop3.2/bin/spark-submit', 'spark_job.py']
subprocess.run(spark_job_command, check=True)

```

2- twitter_listener.py

first we run it as : python3 twitter_listener.py

this scripts is used to listen to twittar api using your token, to extract all the tweets in the interval that contains the word data_engineer , so we can filter it to get the companies hiring.

The fields extracted from this tweet is :

tweet_id : which is id of each tweet

tweet_text : which is the text of the tweet

created_at : which is the time stamp of the tweet

location_name: which is the city that tweet was produced from if determined

language: which is language of the tweet

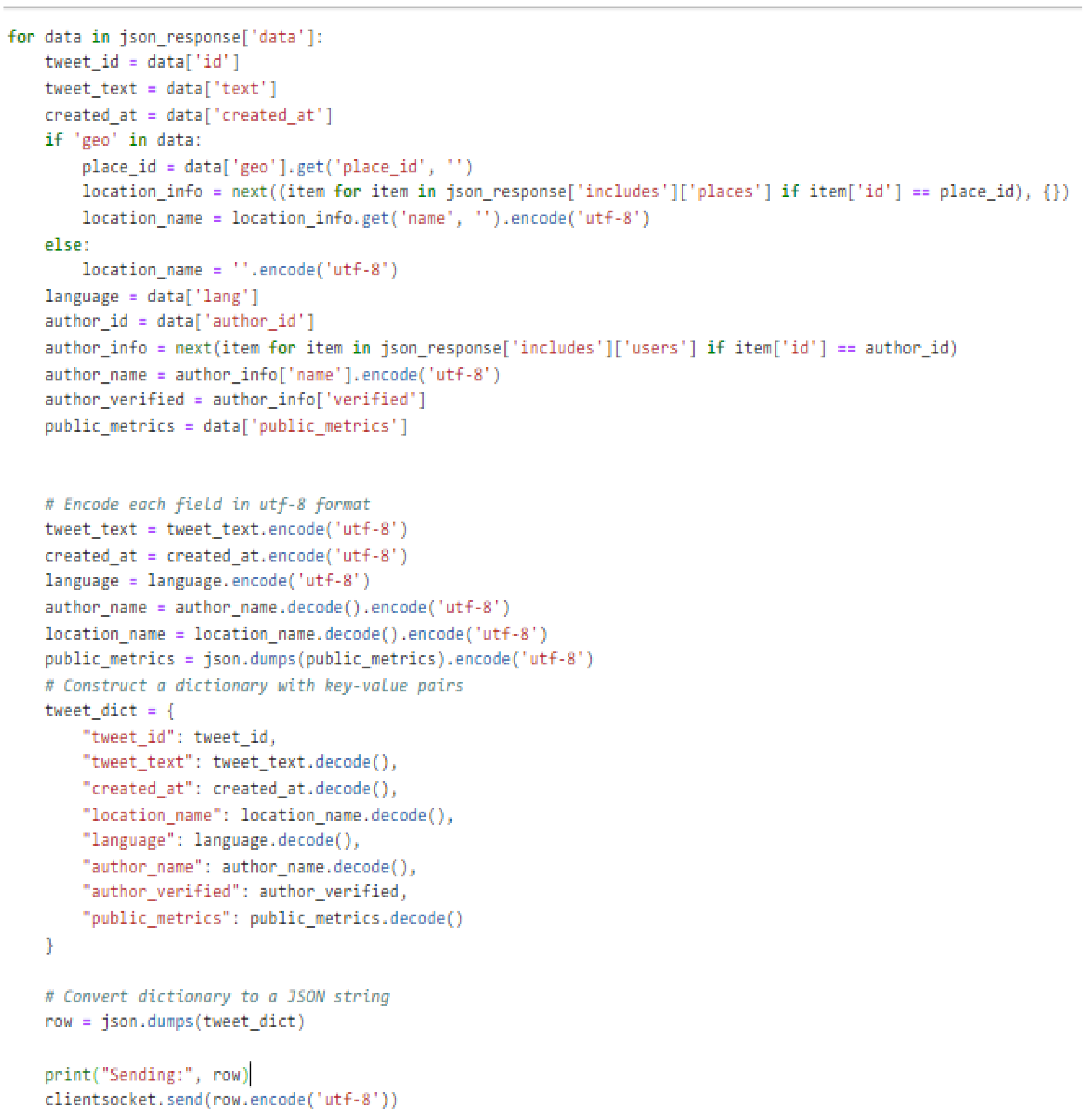
author name : which is the name of the account who wrote tweet

author_verified : which is if the account is verified or not, as if we have many tweets , we can filter for verified as it's more trusted source

public_metrics: which is all metrics of tweet as likes,retweets,etc

we put all those attributes in a forloop and save it in json formate to parse

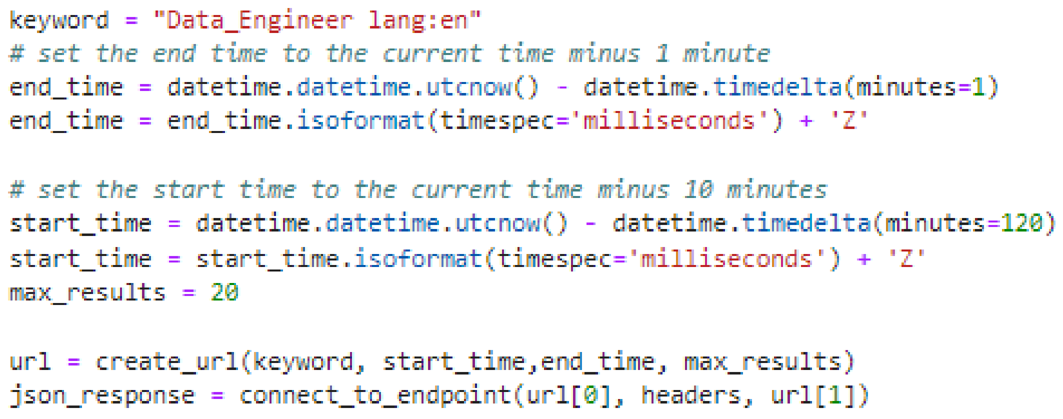
it in the other script to convert it as columns like :



```
for data in json_response['data']:
    tweet_id = data['id']
    tweet_text = data['text']
    created_at = data['created_at']
    if 'geo' in data:
        place_id = data['geo'].get('place_id', '')
        location_info = next((item for item in json_response['includes']['places'] if item['id'] == place_id), {})
        location_name = location_info.get('name', '').encode('utf-8')
    else:
        location_name = ''.encode('utf-8')
    language = data['lang']
    author_id = data['author_id']
    author_info = next(item for item in json_response['includes']['users'] if item['id'] == author_id)
    author_name = author_info['name'].encode('utf-8')
    author_verified = author_info['verified']
    public_metrics = data['public_metrics']


    # Encode each field in utf-8 format
    tweet_text = tweet_text.encode('utf-8')
    created_at = created_at.encode('utf-8')
    language = language.encode('utf-8')
    author_name = author_name.decode().encode('utf-8')
    location_name = location_name.decode().encode('utf-8')
    public_metrics = json.dumps(public_metrics).encode('utf-8')
    # Construct a dictionary with key-value pairs
    tweet_dict = {
        "tweet_id": tweet_id,
        "tweet_text": tweet_text.decode(),
        "created_at": created_at.decode(),
        "location_name": location_name.decode(),
        "language": language.decode(),
        "author_name": author_name.decode(),
        "author_verified": author_verified,
        "public_metrics": public_metrics.decode()
    }

    # Convert dictionary to a JSON string
    row = json.dumps(tweet_dict)

    print("Sending:", row)
    clientsocket.send(row.encode('utf-8'))
```

VIP NOTE: in this listener I wanted to make the start time = the current time – 11 minute, and end time is current time – 1, so if this script runs every 10 minute, it will get all the data of the day, and made maximum 20 to not use a lot of twittar resources, but last 10 minute may or may not have tweets, and may have so few, so for only testing purpose, made start time as current time – 2hours just for testing.

Like:



```
keyword = "Data_Engineer lang:en"
# set the end time to the current time minus 1 minute
end_time = datetime.datetime.utcnow() - datetime.timedelta(minutes=1)
end_time = end_time.isoformat(timespec='milliseconds') + 'Z'

# set the start time to the current time minus 10 minutes
start_time = datetime.datetime.utcnow() - datetime.timedelta(minutes=120)
start_time = start_time.isoformat(timespec='milliseconds') + 'Z'
max_results = 20

url = create_url(keyword, start_time,end_time, max_results)
json_response = connect_to_endpoint(url[0], headers, url[1])
```

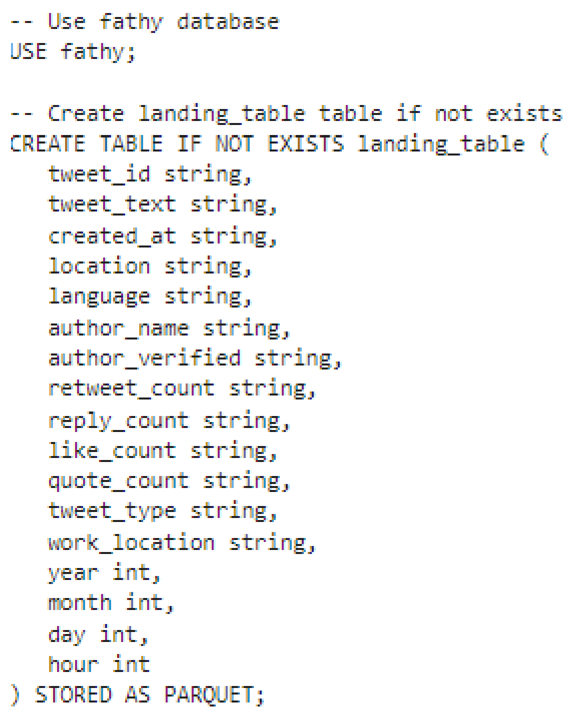
Also didn’t make it always opened because it crashes, in stead made it run for last 10 minute and the script will run every 10 minute of laptop, so like that it gets all data.

3- load_tweet_data

To run this script we use :

hive -f load_tweet_data.hql

this script is so simple, it creates the landing table in hive to load data inside it, don't worry if tables already exist, this script won't make any error because it create database and table only if it doesn't exist.



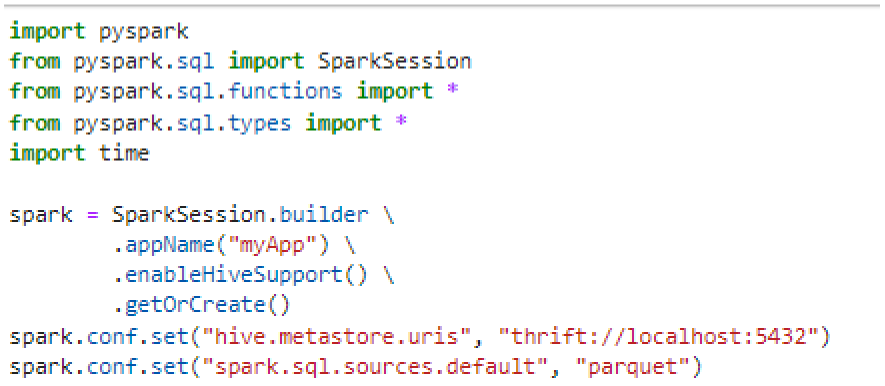
```
-- Use fathy database
USE fathy;

-- Create landing_table table if not exists
CREATE TABLE IF NOT EXISTS landing_table (
   tweet_id string,
   tweet_text string,
   created_at string,
   location string,
   language string,
   author_name string,
   author_verified string,
   retweet_count string,
   reply_count string,
   like_count string,
   quote_count string,
   tweet_type string,
   work_location string,
   year int,
   month int,
   day int,
   hour int
) STORED AS PARQUET;
```

4- twitter_stream.py

to run it : /opt/spark-3.1.2-bin-hadoop3.2/bin/spark-submit twitter_stream.py

that is the main stream that reads data from the twitter listener and clean them to be put into parquet and hive table that we just made in last script.

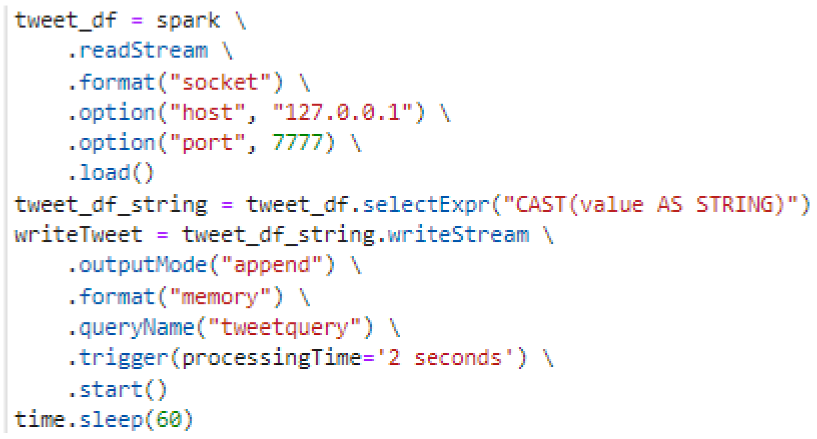
First we open a spark session and set configuration as



```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import time

spark = SparkSession.builder \
        .appName("myApp") \
        .enableHiveSupport() \
        .getOrCreate()
spark.conf.set("hive.metastore.uris", "thrift://localhost:5432")
spark.conf.set("spark.sql.sources.default", "parquet")
```

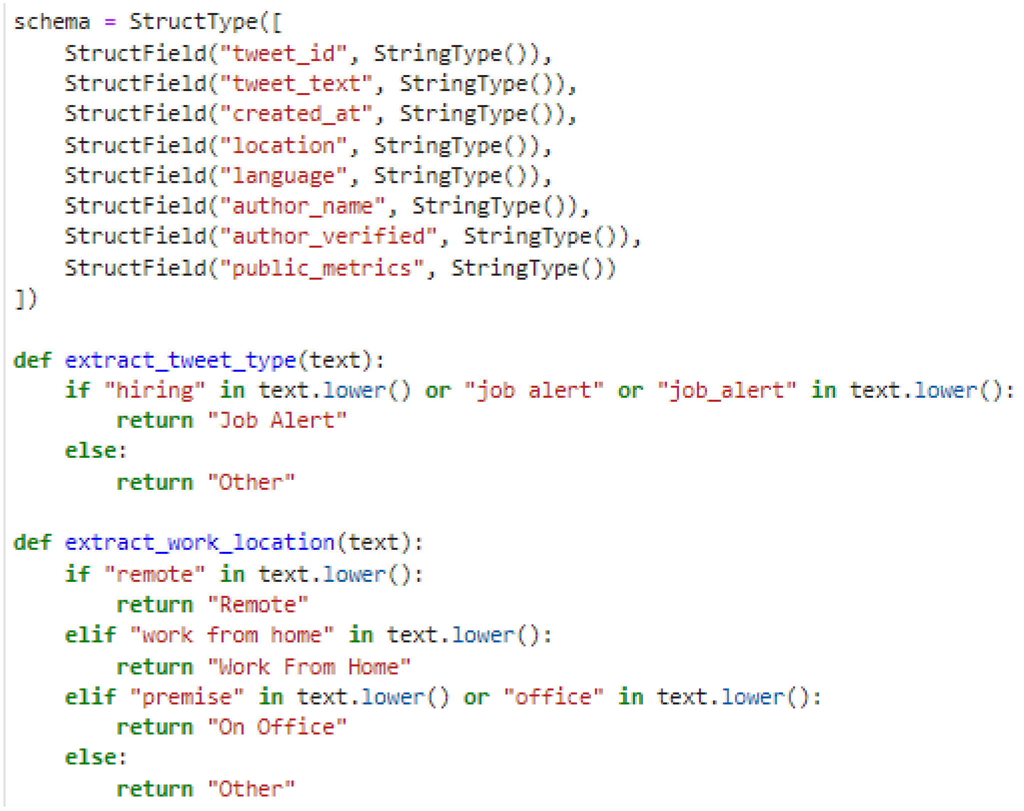
Note: hive for me is location on port 5432, maybe different in other docker machine so please note to change the port that you have from this command : sudo find / -name hive-site.xml and then cat it

Now we will load the port, and connect to twitter_listener through it, then start the streaming and sleep a little until all data come!:



```
tweet_df = spark \
    .readStream \
    .format("socket") \
    .option("host", "127.0.0.1") \
    .option("port", 7777) \
    .load()
tweet_df_string = tweet_df.selectExpr("CAST(value AS STRING)")
writeTweet = tweet_df_string.writeStream \
    .outputMode("append") \
    .format("memory") \
    .queryName("tweetquery") \
    .trigger(processingTime='2 seconds') \
    .start()
time.sleep(60)
```

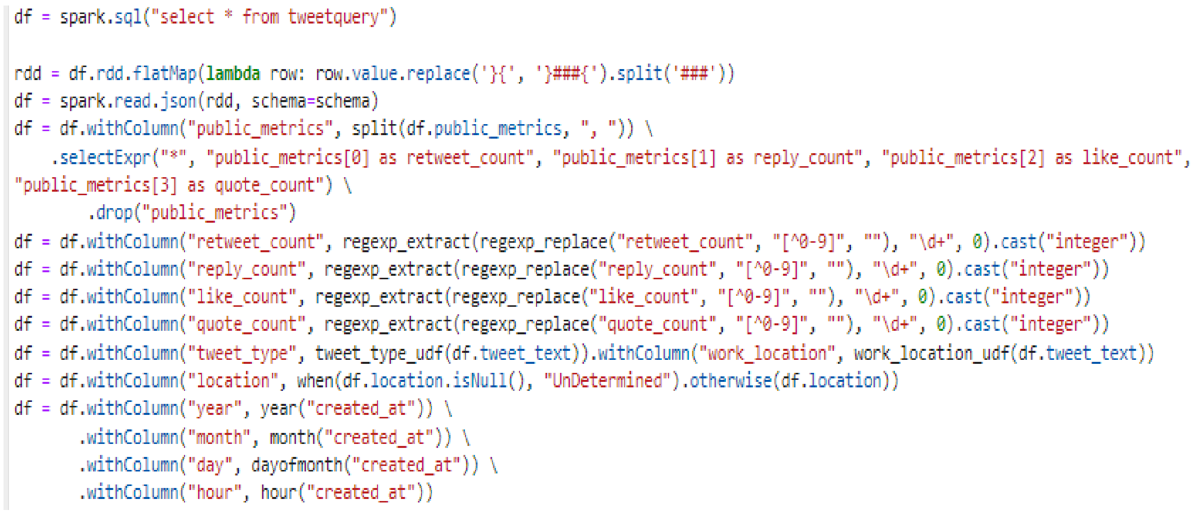
Now made few definitions as we will need it later like the schema of data that will come, function to extract data and put it in another columns like filtering for job_alert tweets and put it in a column



```python
schema = StructType([
    StructField("tweet_id", StringType()),
    StructField("tweet_text", StringType()),
    StructField("created_at", StringType()),
    StructField("location", StringType()),
    StructField("language", StringType()),
    StructField("author_name", StringType()),
    StructField("author_verified", StringType()),
    StructField("public_metrics", StringType())
])

def extract_tweet_type(text):
    if "hiring" in text.lower() or "job alert" or "job_alert" in text.lower():
        return "Job Alert"
    else:
        return "Other"

def extract_work_location(text):
    if "remote" in text.lower():
        return "Remote"
    elif "work from home" in text.lower():
        return "Work From Home"
    elif "premise" in text.lower() or "office" in text.lower():
        return "On Office"
    else:
        return "Other"
```

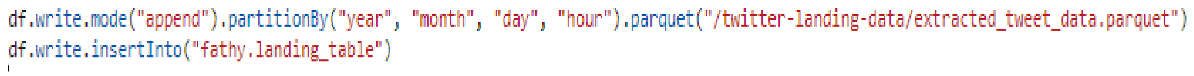
Now we made all transformation we need in data , split the coming data into rows, every row has the columns , then read it into an rdd .

Then :split the created at into 4 columns as year month day hour , also the metrics split everyone into a new column , and applied the functions we just made to search for data in tweet text and insert some certain values.



```
df = spark.sql("select * from tweetquery")

rdd = df.rdd.flatMap(lambda row: row.value.replace('}{', '}###{').split('###'))
df = spark.read.json(rdd, schema=schema)
df = df.withColumn("public_metrics", split(df.public_metrics, ", ")) \
    .selectExpr("*", "public_metrics[0] as retweet_count", "public_metrics[1] as reply_count", "public_metrics[2] as like_count",
"public_metrics[3] as quote_count") \
        .drop("public_metrics")
df = df.withColumn("retweet_count", regexp_extract(regexp_replace("retweet_count", "[^0-9]", ""), "\d+", 0).cast("integer"))
df = df.withColumn("reply_count", regexp_extract(regexp_replace("reply_count", "[^0-9]", ""), "\d+", 0).cast("integer"))
df = df.withColumn("like_count", regexp_extract(regexp_replace("like_count", "[^0-9]", ""), "\d+", 0).cast("integer"))
df = df.withColumn("quote_count", regexp_extract(regexp_replace("quote_count", "[^0-9]", ""), "\d+", 0).cast("integer"))
df = df.withColumn("tweet_type", tweet_type_udf(df.tweet_text)).withColumn("work_location", work_location_udf(df.tweet_text))
df = df.withColumn("location", when(df.location.isNull(), "UnDetermined").otherwise(df.location))
df = df.withColumn("year", year("created_at")) \
       .withColumn("month", month("created_at")) \
       .withColumn("day", dayofmonth("created_at")) \
       .withColumn("hour", hour("created_at"))
```

now we can save it into parquet and load it into hive table we just made:



```
df.write.mode("append").partitionBy("year", "month", "day", "hour").parquet("/twitter-landing-data/extracted_tweet_data.parquet")
df.write.insertInto("fathy.landing_table")
```

and that is the schema of final landing table :

```
root
 |-- tweet_id: string (nullable = true)
 |-- tweet_text: string (nullable = true)
 |-- created_at: string (nullable = true)
 |-- location: string (nullable = true)
 |-- language: string (nullable = true)
 |-- author_name: string (nullable = true)
 |-- author_verified: string (nullable = true)
 |-- retweet_count: integer (nullable = true)
 |-- reply_count: integer (nullable = true)
 |-- like_count: integer (nullable = true)
 |-- quote_count: integer (nullable = true)
 |-- tweet_type: string (nullable = true)
 |-- work_location: string (nullable = true)
 |-- year: integer (nullable = true)
 |-- month: integer (nullable = true)
 |-- day: integer (nullable = true)
 |-- hour: integer (nullable = true)
```

5- create_dimension.hql

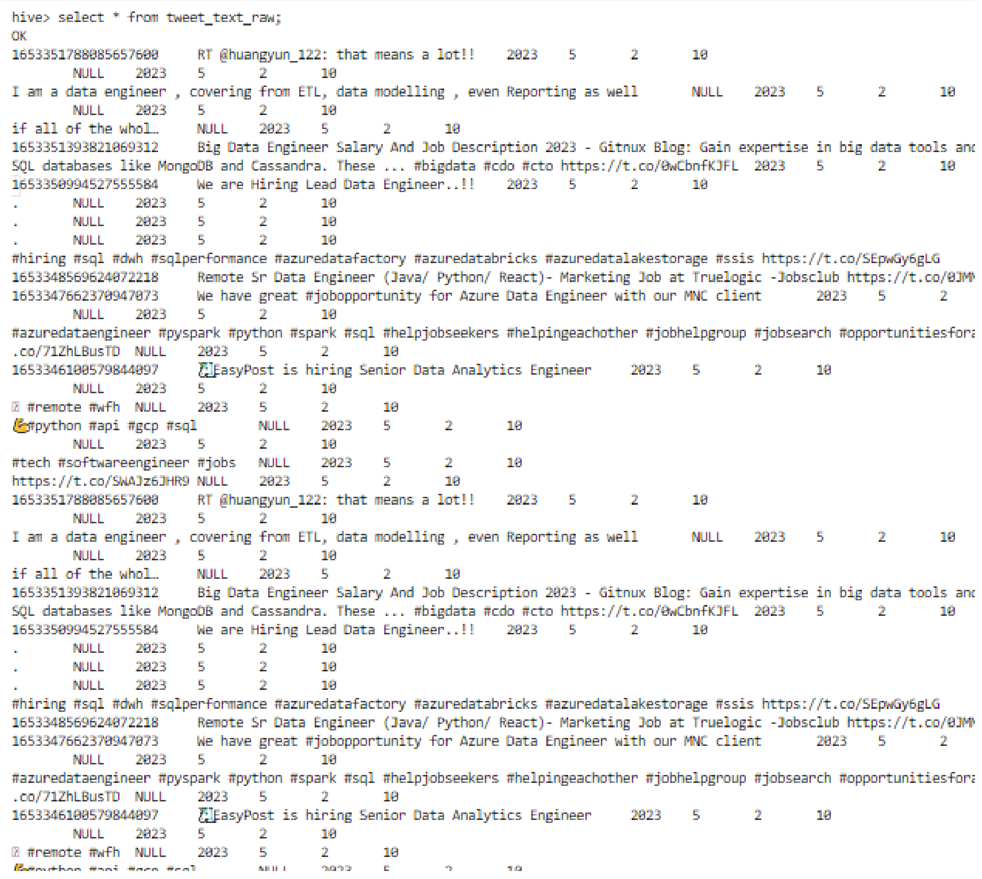
this will create the dimensions it self, this will create the database if doesn't exist called dimension, then insert the required data into the dimension.

Made 4 dimensions:

a) tweet_text_raw : this table is for the text it self of the tweet as:

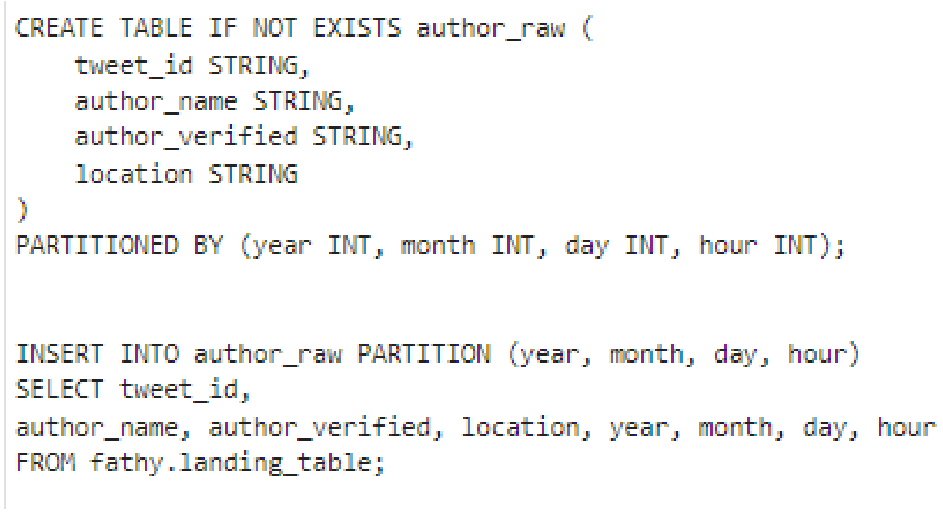
```
CREATE TABLE IF NOT EXISTS tweet_text_raw (
    tweet_id STRING,
    tweet_text STRING
)
PARTITIONED BY (year INT, month INT, day INT, hour INT);

set hive.exec.dynamic.partition.mode=nonstrict;


INSERT INTO TABLE tweet_text_raw PARTITION (year, month, day, hour)
SELECT tweet_id, tweet_text, year, month, day, hour
FROM fathy.landing_table;
```

Data in tweet_text_raw :



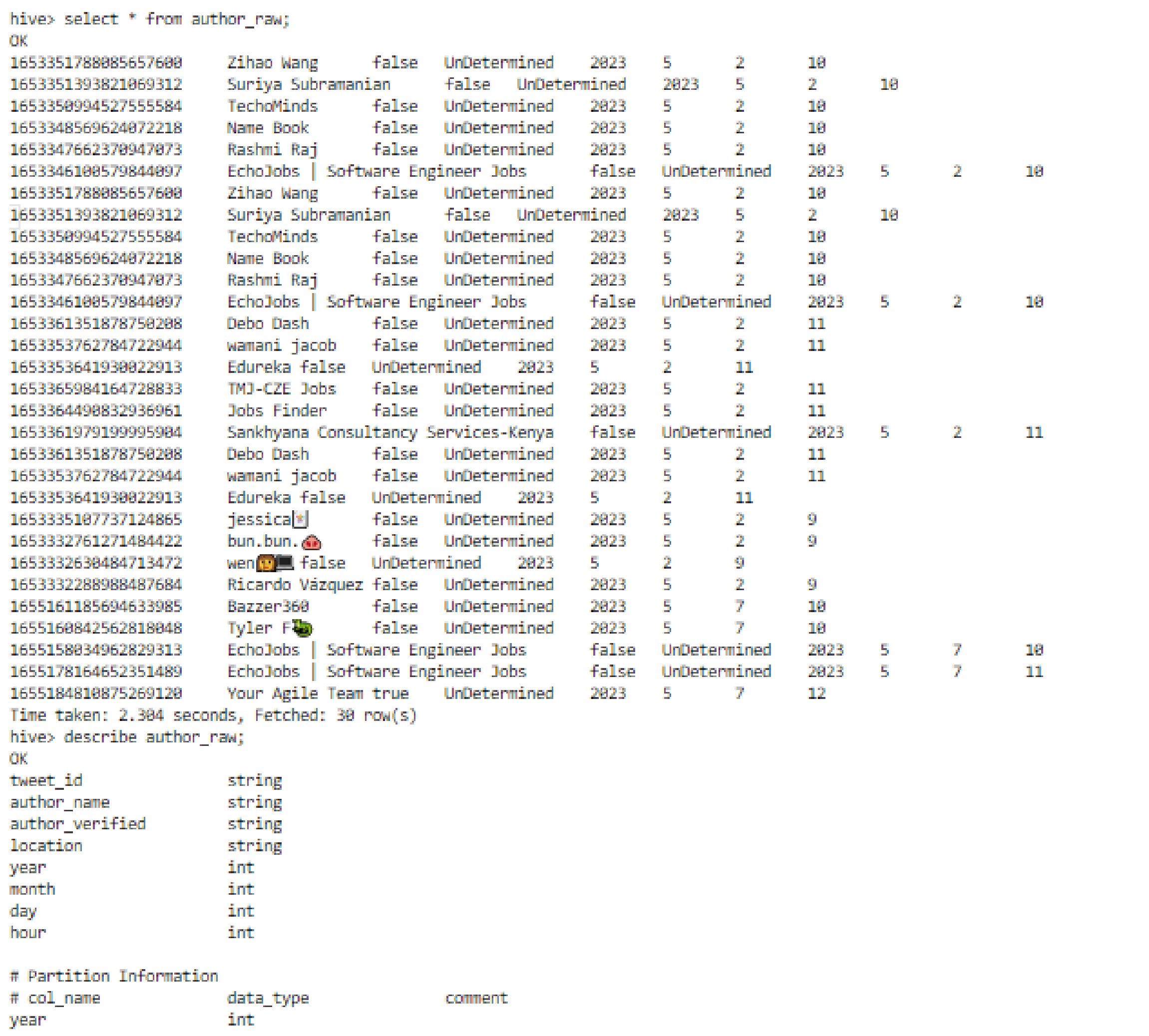
```
hive> select * from tweet_text_raw;
OK
1653351788085657600     RT @huangyun_122: that means a lot!!    2023    5       2       10
        NULL    2023    5       2       10
I am a data engineer , covering from ETL, data modelling , even Reporting as well       NULL    2023    5       2       10
        NULL    2023    5       2       10
if all of the whol…     NULL    2023    5       2       10
1653351393821069312     Big Data Engineer Salary And Job Description 2023 - Gitnux Blog: Gain expertise in big data tools anc
SQL databases like MongoDB and Cassandra. These ... #bigdata #cdo #cto https://t.co/0wCbnfKJFL  2023    5       2       10
1653350994527555584     We are Hiring Lead Data Engineer..!!    2023    5       2       10
.       NULL    2023    5       2       10
.       NULL    2023    5       2       10
.       NULL    2023    5       2       10
#hiring #sql #dwh #sqlperformance #azuredatafactory #azuredatabricks #azuredatalakestorage #ssis https://t.co/SEpwGy6gLG
1653348569624072218     Remote Sr Data Engineer (Java/ Python/ React)- Marketing Job at Truelogic -Jobsclub https://t.co/0JM
1653347662370947073     We have great #jobopportunity for Azure Data Engineer with our MNC client       2023    5       2
        NULL    2023    5       2       10
#azuredataengineer #pyspark #python #spark #sql #helpjobseekers #helpingeachother #jobhelpgroup #jobsearch #opportunitiesfor
.co/71ZhLBusTD  NULL    2023    5       2       10
1653346100579844097     EasyPost is hiring Senior Data Analytics Engineer      2023    5       2       10
        NULL    2023    5       2       10
 #remote #wfh   NULL    2023    5       2       10
#python #api #gcp #sql          NULL    2023    5       2       10
        NULL    2023    5       2       10
#tech #softwareengineer #jobs   NULL    2023    5       2       10
https://t.co/SWAJz6JHR9 NULL    2023    5       2       10
1653351788085657600     RT @huangyun_122: that means a lot!!    2023    5       2       10
        NULL    2023    5       2       10
I am a data engineer , covering from ETL, data modelling , even Reporting as well       NULL    2023    5       2       10
        NULL    2023    5       2       10
if all of the whol…     NULL    2023    5       2       10
1653351393821069312     Big Data Engineer Salary And Job Description 2023 - Gitnux Blog: Gain expertise in big data tools anc
SQL databases like MongoDB and Cassandra. These ... #bigdata #cdo #cto https://t.co/0wCbnfKJFL  2023    5       2       10
1653350994527555584     We are Hiring Lead Data Engineer..!!    2023    5       2       10
.       NULL    2023    5       2       10
.       NULL    2023    5       2       10
.       NULL    2023    5       2       10
#hiring #sql #dwh #sqlperformance #azuredatafactory #azuredatabricks #azuredatalakestorage #ssis https://t.co/SEpwGy6gLG
1653348569624072218     Remote Sr Data Engineer (Java/ Python/ React)- Marketing Job at Truelogic -Jobsclub https://t.co/0JM
1653347662370947073     We have great #jobopportunity for Azure Data Engineer with our MNC client       2023    5       2
        NULL    2023    5       2       10
#azuredataengineer #pyspark #python #spark #sql #helpjobseekers #helpingeachother #jobhelpgroup #jobsearch #opportunitiesfor
.co/71ZhLBusTD  NULL    2023    5       2       10
1653346100579844097     EasyPost is hiring Senior Data Analytics Engineer      2023    5       2       10
        NULL    2023    5       2       10
 #remote #wfh   NULL    2023    5       2       10
#python #api #gcp #sql          NULL    2023    5       2       10
```

b) author_raw : this dimension will have the data for author as:



```
CREATE TABLE IF NOT EXISTS author_raw (
    tweet_id STRING,
    author_name STRING,
    author_verified STRING,
    location STRING
)
PARTITIONED BY (year INT, month INT, day INT, hour INT);


INSERT INTO author_raw PARTITION (year, month, day, hour)
SELECT tweet_id,
author_name, author_verified, location, year, month, day, hour
FROM fathy.landing_table;
```

Data in author_raw :

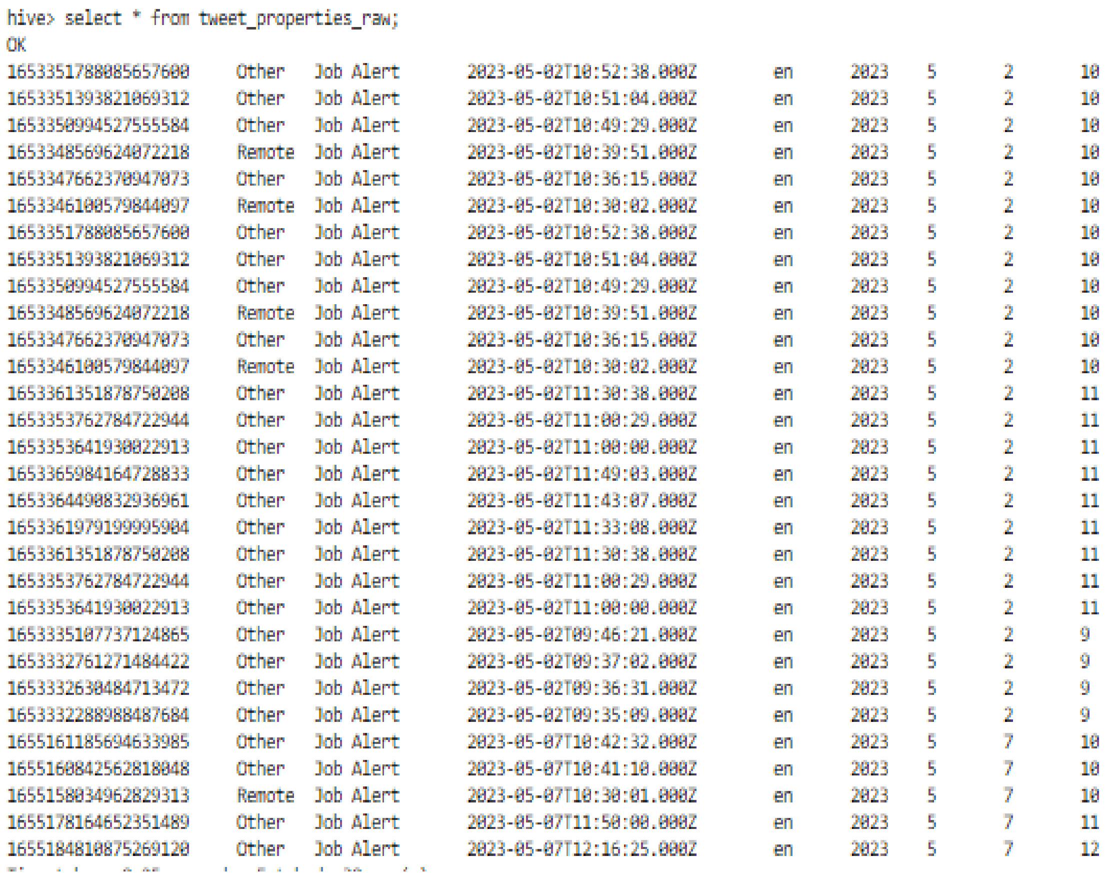


```
hive> select * from author_raw;
OK
1653351788085657600     Zihao Wang      false   UnDetermined    2023    5       2       10
1653351393821069312     Suriya Subramanian      false   UnDetermined    2023    5       2       10
1653350994527555584     TechoMinds      false   UnDetermined    2023    5       2       10
1653348569624072218     Name Book       false   UnDetermined    2023    5       2       10
1653347662370947073     Rashmi Raj      false   UnDetermined    2023    5       2       10
1653346100579844097     EchoJobs | Software Engineer Jobs       false   UnDetermined    2023    5       2       10
1653351788085657600     Zihao Wang      false   UnDetermined    2023    5       2       10
1653351393821069312     Suriya Subramanian      false   UnDetermined    2023    5       2       10
1653350994527555584     TechoMinds      false   UnDetermined    2023    5       2       10
1653348569624072218     Name Book       false   UnDetermined    2023    5       2       10
1653347662370947073     Rashmi Raj      false   UnDetermined    2023    5       2       10
1653346100579844097     EchoJobs | Software Engineer Jobs       false   UnDetermined    2023    5       2       10
1653361351878750208     Debo Dash       false   UnDetermined    2023    5       2       11
1653353762784722944     wamani jacob    false   UnDetermined    2023    5       2       11
1653353641930022913     Edureka false   UnDetermined    2023    5       2       11
1653365984164728833     TMJ-CZE Jobs    false   UnDetermined    2023    5       2       11
1653364490832936961     Jobs Finder     false   UnDetermined    2023    5       2       11
1653361979199995904     Sankhyana Consultancy Services-Kenya    false   UnDetermined    2023    5       2       11
1653361351878750208     Debo Dash       false   UnDetermined    2023    5       2       11
1653353762784722944     wamani jacob    false   UnDetermined    2023    5       2       11
1653353641930022913     Edureka false   UnDetermined    2023    5       2       11
1653335107737124865     jessica🃏       false   UnDetermined    2023    5       2       9
1653332761271484422     bun.bun.🍄      false   UnDetermined    2023    5       2       9
1653332630484713472     wen👩‍💻 false   UnDetermined    2023    5       2       9
1653332288988487684     Ricardo Vázquez false   UnDetermined    2023    5       2       9
1655161185694633985     Bazzer360       false   UnDetermined    2023    5       7       10
1655160842562818048     Tyler F🐢       false   UnDetermined    2023    5       7       10
1655158034962829313     EchoJobs | Software Engineer Jobs       false   UnDetermined    2023    5       7       10
1655178164652351489     EchoJobs | Software Engineer Jobs       false   UnDetermined    2023    5       7       11
1655184810875269120     Your Agile Team true    UnDetermined    2023    5       7       12
Time taken: 2.304 seconds, Fetched: 30 row(s)
hive> describe author_raw;
OK
tweet_id                string
author_name             string
author_verified         string
location                string
year                    int
month                   int
day                     int
hour                    int

# Partition Information
# col_name              data_type               comment
year                    int
```

c) tweet_properties_raw: this will have some information about tweet

```
CREATE TABLE IF NOT EXISTS tweet_properties_raw (
    tweet_id STRING,
    work_location STRING,
    tweet_type STRING,
    created_at STRING,
    language STRING
)
PARTITIONED BY (year INT, month INT, day INT, hour INT);


INSERT INTO tweet_properties_raw PARTITION (year, month, day, hour)
       SELECT tweet_id,
       work_location,
       tweet_type,
       created_at,
       language,
       year,
       month,
       day,
       hour
FROM fathy.landing_table;
```

Data in tweet_properties_raw:

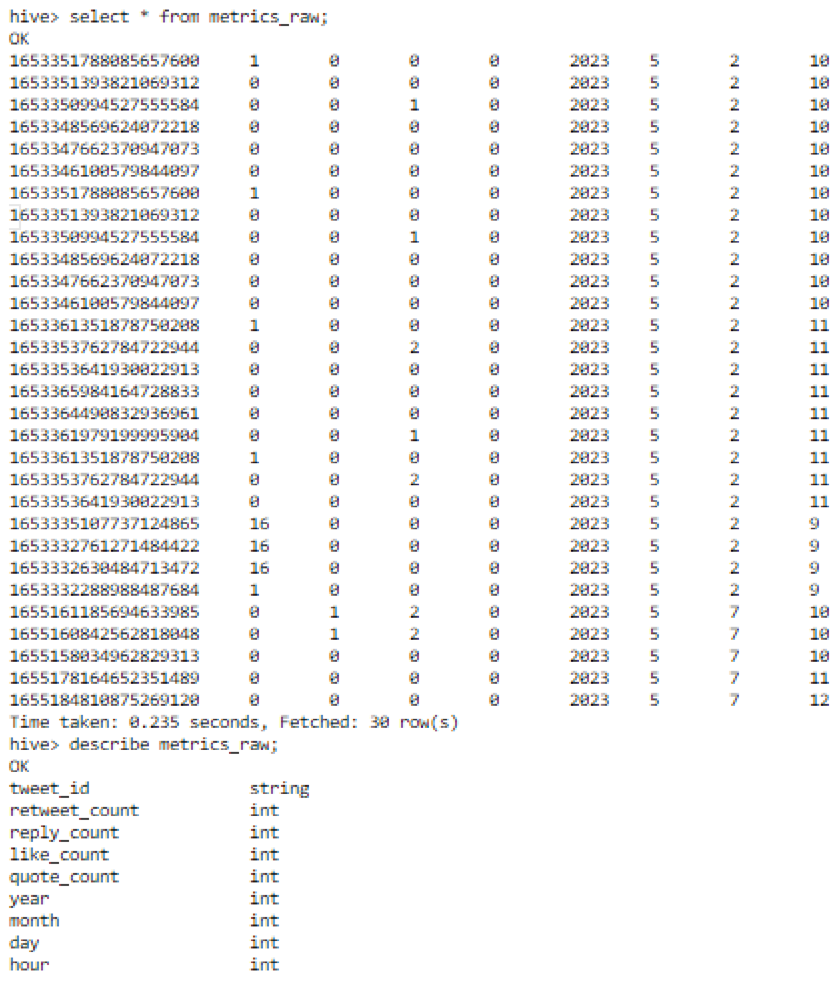


```
hive> select * from tweet_properties_raw;
OK
1653351788085657600     Other   Job Alert       2023-05-02T10:52:38.000Z        en      2023    5       2       10
1653351393821069312     Other   Job Alert       2023-05-02T10:51:04.000Z        en      2023    5       2       10
1653350994527555584     Other   Job Alert       2023-05-02T10:49:29.000Z        en      2023    5       2       10
1653348569624072218     Remote  Job Alert       2023-05-02T10:39:51.000Z        en      2023    5       2       10
1653347662370947073     Other   Job Alert       2023-05-02T10:36:15.000Z        en      2023    5       2       10
1653346100579844097     Remote  Job Alert       2023-05-02T10:30:02.000Z        en      2023    5       2       10
1653351788085657600     Other   Job Alert       2023-05-02T10:52:38.000Z        en      2023    5       2       10
1653351393821069312     Other   Job Alert       2023-05-02T10:51:04.000Z        en      2023    5       2       10
1653350994527555584     Other   Job Alert       2023-05-02T10:49:29.000Z        en      2023    5       2       10
1653348569624072218     Remote  Job Alert       2023-05-02T10:39:51.000Z        en      2023    5       2       10
1653347662370947073     Other   Job Alert       2023-05-02T10:36:15.000Z        en      2023    5       2       10
1653346100579844097     Remote  Job Alert       2023-05-02T10:30:02.000Z        en      2023    5       2       10
1653361351878750208     Other   Job Alert       2023-05-02T11:30:38.000Z        en      2023    5       2       11
1653353762784722944     Other   Job Alert       2023-05-02T11:00:29.000Z        en      2023    5       2       11
1653353641930022913     Other   Job Alert       2023-05-02T11:00:00.000Z        en      2023    5       2       11
1653365984164728833     Other   Job Alert       2023-05-02T11:49:03.000Z        en      2023    5       2       11
1653364490832936961     Other   Job Alert       2023-05-02T11:43:07.000Z        en      2023    5       2       11
1653361979199995904     Other   Job Alert       2023-05-02T11:33:08.000Z        en      2023    5       2       11
1653361351878750208     Other   Job Alert       2023-05-02T11:30:38.000Z        en      2023    5       2       11
1653353762784722944     Other   Job Alert       2023-05-02T11:00:29.000Z        en      2023    5       2       11
1653353641930022913     Other   Job Alert       2023-05-02T11:00:00.000Z        en      2023    5       2       11
1653335107737124865     Other   Job Alert       2023-05-02T09:46:21.000Z        en      2023    5       2       9
1653332761271484422     Other   Job Alert       2023-05-02T09:37:02.000Z        en      2023    5       2       9
1653332630484713472     Other   Job Alert       2023-05-02T09:36:31.000Z        en      2023    5       2       9
1653332288988487684     Other   Job Alert       2023-05-02T09:35:09.000Z        en      2023    5       2       9
1655161185694633985     Other   Job Alert       2023-05-07T10:42:32.000Z        en      2023    5       7       10
1655160842562818048     Other   Job Alert       2023-05-07T10:41:10.000Z        en      2023    5       7       10
1655158034962829313     Remote  Job Alert       2023-05-07T10:30:01.000Z        en      2023    5       7       10
1655178164652351489     Other   Job Alert       2023-05-07T11:50:00.000Z        en      2023    5       7       11
1655184810875269120     Other   Job Alert       2023-05-07T12:16:25.000Z        en      2023    5       7       12
```

d) metrics_raw : which has data inside metrics_raw

```
CREATE TABLE IF NOT EXISTS metrics_raw (
    tweet_id STRING,
    retweet_count INT,
    reply_count INT,
    like_count INT,
    quote_count INT
)
PARTITIONED BY (year INT, month INT, day INT, hour INT);


INSERT INTO metrics_raw PARTITION (year, month, day, hour)
       SELECT tweet_id,
       retweet_count,
       reply_count,
       like_count,
       quote_count,
       year,
       month ,
       day ,
       hour
FROM fathy.landing_table;
```

Data in metrics_raw:



```
hive> select * from metrics_raw;
OK
1653351788085657600     1       0       0       0       2023    5       2       10
1653351393821069312     0       0       0       0       2023    5       2       10
1653350994527555584     0       0       1       0       2023    5       2       10
1653348569624072218     0       0       0       0       2023    5       2       10
1653347662370947073     0       0       0       0       2023    5       2       10
1653346100579844097     0       0       0       0       2023    5       2       10
1653351788085657600     1       0       0       0       2023    5       2       10
1653351393821069312     0       0       0       0       2023    5       2       10
1653350994527555584     0       0       1       0       2023    5       2       10
1653348569624072218     0       0       0       0       2023    5       2       10
1653347662370947073     0       0       0       0       2023    5       2       10
1653346100579844097     0       0       0       0       2023    5       2       10
1653361351878750208     1       0       0       0       2023    5       2       11
1653353762784722944     0       0       2       0       2023    5       2       11
1653353641930022913     0       0       0       0       2023    5       2       11
1653365984164728833     0       0       0       0       2023    5       2       11
1653364490832936961     0       0       0       0       2023    5       2       11
1653361979199995904     0       0       1       0       2023    5       2       11
1653361351878750208     1       0       0       0       2023    5       2       11
1653353762784722944     0       0       2       0       2023    5       2       11
1653353641930022913     0       0       0       0       2023    5       2       11
1653335107737124865     16      0       0       0       2023    5       2       9
1653332761271484422     16      0       0       0       2023    5       2       9
1653332630484713472     16      0       0       0       2023    5       2       9
1653332288988487684     1       0       0       0       2023    5       2       9
1655161185694633985     0       1       2       0       2023    5       7       10
1655160842562818048     0       1       2       0       2023    5       7       10
1655158034962829313     0       0       0       0       2023    5       7       10
1655178164652351489     0       0       0       0       2023    5       7       11
1655184810875269120     0       0       0       0       2023    5       7       12
Time taken: 0.235 seconds, Fetched: 30 row(s)
hive> describe metrics_raw;
OK
tweet_id                string
retweet_count           int
reply_count             int
like_count              int
quote_count             int
year                    int
month                   int
day                     int
hour                    int
```

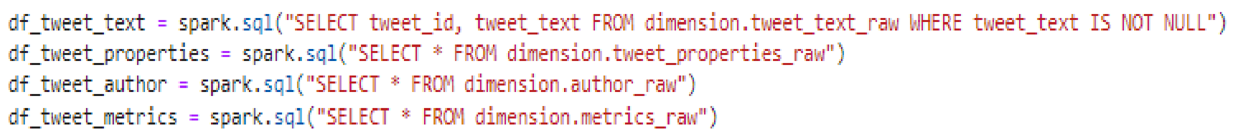
Now all dimensions are ready to make the fact table!

6- spark_job

to run it: /opt/spark-3.1.2-bin-hadoop3.2/bin/spark-submit spark_job.py

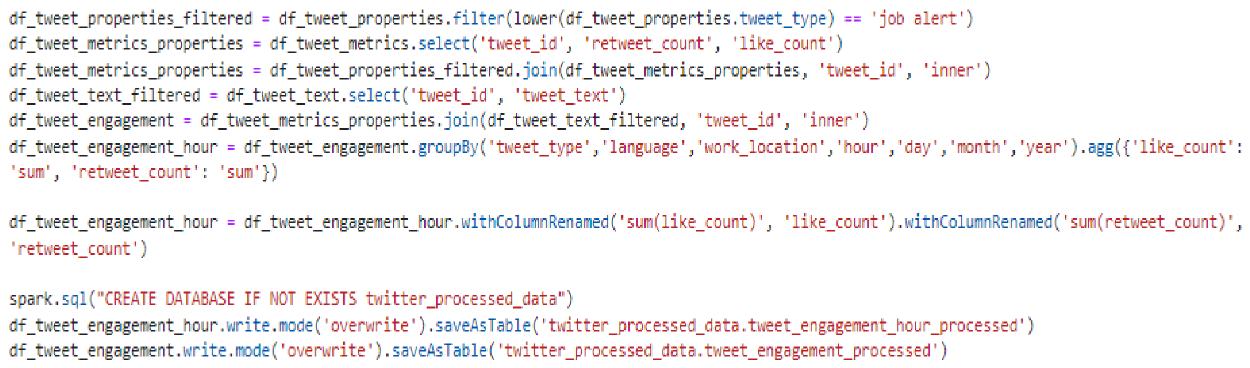
this will read the data from hive's dimension and make fact tables

first read files from hive as :



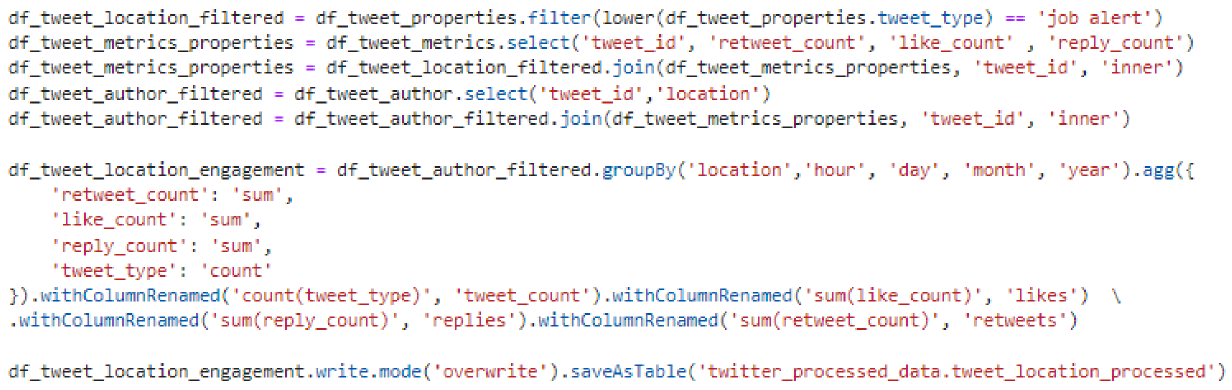
```
df_tweet_text = spark.sql("SELECT tweet_id, tweet_text FROM dimension.tweet_text_raw WHERE tweet_text IS NOT NULL")
df_tweet_properties = spark.sql("SELECT * FROM dimension.tweet_properties_raw")
df_tweet_author = spark.sql("SELECT * FROM dimension.author_raw")
df_tweet_metrics = spark.sql("SELECT * FROM dimension.metrics_raw")
```

now we will make fact tables from this hive as:



```
df_tweet_properties_filtered = df_tweet_properties.filter(lower(df_tweet_properties.tweet_type) == 'job alert')
df_tweet_metrics_properties = df_tweet_metrics.select('tweet_id', 'retweet_count', 'like_count')
df_tweet_metrics_properties = df_tweet_properties_filtered.join(df_tweet_metrics_properties, 'tweet_id', 'inner')
df_tweet_text_filtered = df_tweet_text.select('tweet_id', 'tweet_text')
df_tweet_engagement = df_tweet_metrics_properties.join(df_tweet_text_filtered, 'tweet_id', 'inner')
df_tweet_engagement_hour = df_tweet_engagement.groupBy('tweet_type','language','work_location','hour','day','month','year').agg({'like_count':
'sum', 'retweet_count': 'sum'})

df_tweet_engagement_hour = df_tweet_engagement_hour.withColumnRenamed('sum(like_count)', 'like_count').withColumnRenamed('sum(retweet_count)',
'retweet_count')

spark.sql("CREATE DATABASE IF NOT EXISTS twitter_processed_data")
df_tweet_engagement_hour.write.mode('overwrite').saveAsTable('twitter_processed_data.tweet_engagement_hour_processed')
df_tweet_engagement.write.mode('overwrite').saveAsTable('twitter_processed_data.tweet_engagement_processed')
```

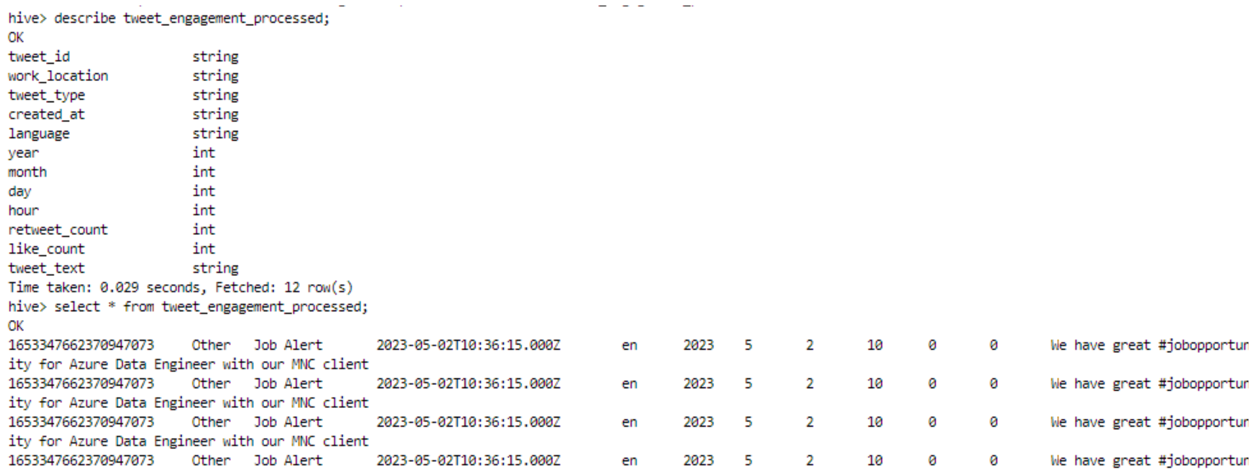
then made another fact table as :



```
df_tweet_location_filtered = df_tweet_properties.filter(lower(df_tweet_properties.tweet_type) == 'job alert')
df_tweet_metrics_properties = df_tweet_metrics.select('tweet_id', 'retweet_count', 'like_count' , 'reply_count')
df_tweet_metrics_properties = df_tweet_location_filtered.join(df_tweet_metrics_properties, 'tweet_id', 'inner')
df_tweet_author_filtered = df_tweet_author.select('tweet_id','location')
df_tweet_author_filtered = df_tweet_author_filtered.join(df_tweet_metrics_properties, 'tweet_id', 'inner')

df_tweet_location_engagement = df_tweet_author_filtered.groupBy('location','hour', 'day', 'month', 'year').agg({
    'retweet_count': 'sum',
    'like_count': 'sum',
    'reply_count': 'sum',
    'tweet_type': 'count'
}).withColumnRenamed('count(tweet_type)', 'tweet_count').withColumnRenamed('sum(like_count)', 'likes')  \
.withColumnRenamed('sum(reply_count)', 'replies').withColumnRenamed('sum(retweet_count)', 'retweets')

df_tweet_location_engagement.write.mode('overwrite').saveAsTable('twitter_processed_data.tweet_location_processed')
```

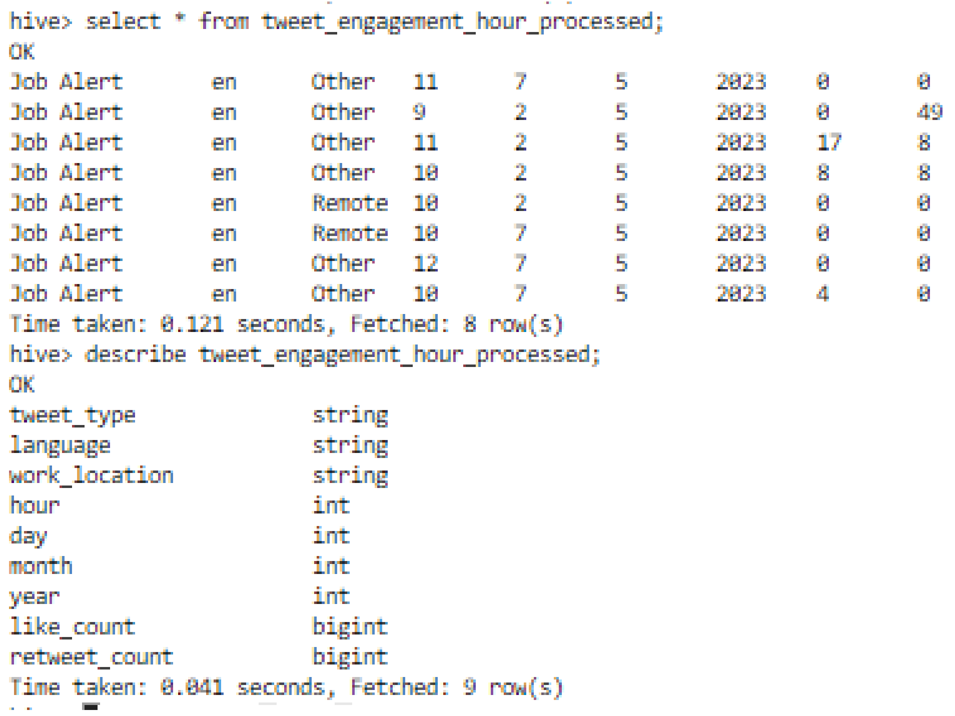
Now let's take a look on the data it self for each fact table as :

1) tweet_engagement_processed :
   this fact table tweets that is only looking for jobs like id of tweet and it's properties, like language? Timestamp? When?Likes? Retweets? And so on to make queries on it filtered for job alert
   and here sample of data:



```
hive> describe tweet_engagement_processed;
OK
tweet_id                string
work_location           string
tweet_type              string
created_at              string
language                string
year                    int
month                   int
day                     int
hour                    int
retweet_count           int
like_count              int
tweet_text              string
Time taken: 0.029 seconds, Fetched: 12 row(s)
hive> select * from tweet_engagement_processed;
OK
1653347662370947073     Other   Job Alert       2023-05-02T10:36:15.000Z        en      2023    5       2       10      0       0       We have great #jobopportur
ity for Azure Data Engineer with our MNC client
1653347662370947073     Other   Job Alert       2023-05-02T10:36:15.000Z        en      2023    5       2       10      0       0       We have great #jobopportur
ity for Azure Data Engineer with our MNC client
1653347662370947073     Other   Job Alert       2023-05-02T10:36:15.000Z        en      2023    5       2       10      0       0       We have great #jobopportur
ity for Azure Data Engineer with our MNC client
1653347662370947073     Other   Job Alert       2023-05-02T10:36:15.000Z        en      2023    5       2       10      0       0       We have great #jobopportur
```

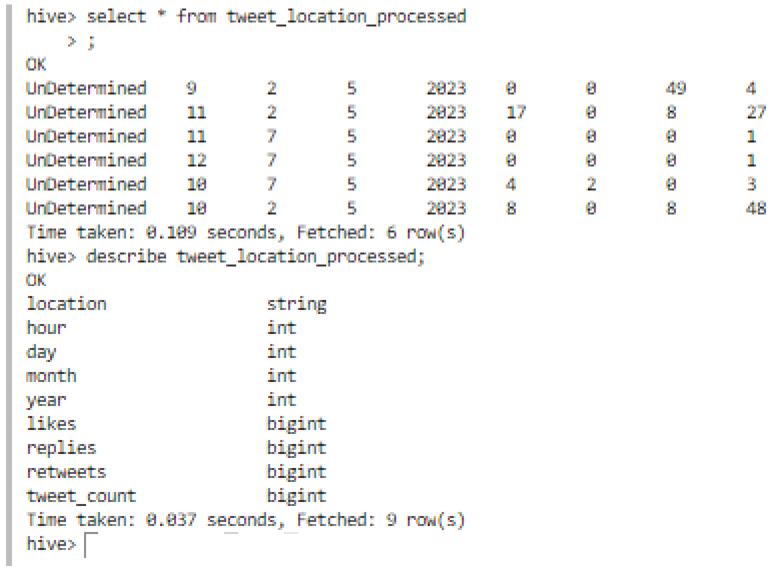
2) tweet_engagement_hour_processed
Same table as upove but grouped by hour/day/month/year to get more aggregated data



```
hive> select * from tweet_engagement_hour_processed;
OK
Job Alert       en      Other   11      7       5       2023    0       0
Job Alert       en      Other   9       2       5       2023    0       49
Job Alert       en      Other   11      2       5       2023    17      8
Job Alert       en      Other   10      2       5       2023    8       8
Job Alert       en      Remote  10      2       5       2023    0       0
Job Alert       en      Remote  10      7       5       2023    0       0
Job Alert       en      Other   12      7       5       2023    0       0
Job Alert       en      Other   10      7       5       2023    4       0
Time taken: 0.121 seconds, Fetched: 8 row(s)
hive> describe tweet_engagement_hour_processed;
OK
tweet_type              string
language                string
work_location           string
hour                    int
day                     int
month                   int
year                    int
like_count              bigint
retweet_count           bigint
Time taken: 0.041 seconds, Fetched: 9 row(s)
```

3) now 3rd fact table which is tweet_egagement_location_processed;

which groups for every country the tweets, and if it's Undetermined it says undetermined means twitter couldn't fetch the country of the person who made the tweet.
As my data set here is for only last 10minute so I don't have much data, so I got only undetermined, but if a lot of data should be different countries as twittar doesn't allow giving location to developers unless they allow it .



```
hive> select * from tweet_location_processed
    > ;
OK
UnDetermined    9       2       5       2023    0       0       49      4
UnDetermined    11      2       5       2023    17      0       8       27
UnDetermined    11      7       5       2023    0       0       0       1
UnDetermined    12      7       5       2023    0       0       0       1
UnDetermined    10      7       5       2023    4       2       0       3
UnDetermined    10      2       5       2023    8       0       8       48
Time taken: 0.109 seconds, Fetched: 6 row(s)
hive> describe tweet_location_processed;
OK
location                string
hour                    int
day                     int
month                   int
year                    int
likes                   bigint
replies                 bigint
retweets                bigint
tweet_count             bigint
Time taken: 0.037 seconds, Fetched: 9 row(s)
hive>
```

Now will make cron job to run this script every 10 minutes:
Now write command:
a- crontab -e
then write that line inside it :
*/10 * * * * python3 /home/itversity/itversity-material/project/twitter_pipeline.py >/dev/null 2>&1

This will run all the pipline every 10 minute to extract data in last 10 minute